Journal of Network and Computer Applications 187 (2021) 103108

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Automated DDOS attack detection in software defined networking

Nisha Ahuja [a], Gaurav Singal [a], Debajyoti Mukhopadhyay [a], Neeraj Kumar [b,c,*]

[a] Department of Computer Science Engineering, SEAS, Bennett University, Greater Noida, India
[b] Thapar Institute of Engineering and Technology, India and Department of Computer Science and Information Engineering, Asia University, Taiwan and King Abdul Aziz University, Saudi Arabia
[c] School of Computer Science, University of Petroleum and Energy Studies, Dehradun, Utttarakhand, India

ARTICLE INFO

Keywords:
Machine learning
Software-defined-networking
DDOS Attack detection
Traffic protocol
SDN Traffic classification
Mininet

MSC:
00–01
99-00

ABSTRACT

Software-Defined Networking (SDN) is a networking paradigm that has redefined the term network by making the network devices programmable. SDN helps network engineers to monitor the network expeditely, control the network from a central point, identify malicious traffic and link failure in easy and efficient manner. Besides such flexibility provided by SDN, it is also vulnerable to attacks such as DDoS which can halt the complete network. To mitigate this attack, the paper proposes to classify the benign traffic from DDoS attack traffic by using machine learning technique. The major contribution of this paper is identification of novel features for DDoS attack detections. Novel features are logged into CSV file to create the dataset and machine learning algorithms are trained on the created SDN dataset. Various work which has already been done for DDoS attack detection either used a non-SDN dataset or the research data is not made public. A novel hybrid machine learning model is utilized to perform the classification. Results show that the hybrid model of Support Vector classifier with Random Forest (SVC-RF) classifies the traffic with the highest testing accuracy of 98.8% with a very low false alarm rate.

1. Introduction

Software-Defined Networking(SDN) (Dabbagh et al., 2015) is the networking architecture defined by the software program. In SDN, network traffic is controlled by the software which directs the traffic between the hosts. This is unlike the current network architecture where the traffic control is hardware-based which is defined by the switches, routers and, other network infrastructure. The centralized SDN architecture is designed in such a way that SDN switch is comprised of only the data plane and the control plane is moved to another entity known as the controller. The controller acts as the brain of the network and it is a networkwide centralized entity where all the switches in the network abide by the decision made by the controller. Examples of SDN controllers include FloodLight, Ryu (Natarajan, 2014a), Pox, OpenDayLight (Garg et al., 2019), Nox, etc. which are open source and incorporates a set of APIs for building network applications.

SDN solves many of the challenges of the current network like Dynamic configuration (Kreutz et al., 2013) which implies at run time the network can be configured remotely, Vendor independence in choosing the network devices which means the switches can be chosen independent of the vendor as any requirement can be built into it via programming, Run-time network management, Cheaper network devices which implies that the switches contain only the dumb data as expensive control plane circuitry is moved to a single central location which leads to a reduction in price, improved QoS (Singal et al., 2017a, 2017b) and, the ability to identify link failure (Varadharajan et al., 2018).

Apart from the various advantages offered by SDN, it suffers from security issues which can be exploited in the different architectural planes (Wang et al., 2015a), (Yoon et al., 2017). In this paper, we are going to classify benign and Distributed Denial of Service attack (DDOS) traffic with the help of machine learning techniques (Said Elsayed et al., 1910).

In DDOS attack (Kiwonet al., 2017) presented in Fig. 1, the goal of the attacker is to overflow the resources of the target host to interrupt the benign host. The DDoS attack in SDN (Shin and Gu, 2013) can take place at different architectural planes is discuss below:

- Application Plane attack: The attack occurs through the applications present in the application plane. The malicious application exhausts the resources so that legitimate users suffer.

* Corresponding author.
E-mail addresses: na6742@bennett.edu.in (N. Ahuja), gauravsingal789@gmail.com (G. Singal), debajyoti.mukhopadhyay@gmail.com (D. Mukhopadhyay), neeraj.kumar@thapar.edu (N. Kumar).

https://doi.org/10.1016/j.jnca.2021.103108
Received 9 November 2020; Received in revised form 12 March 2021; Accepted 7 May 2021
Available online 19 May 2021

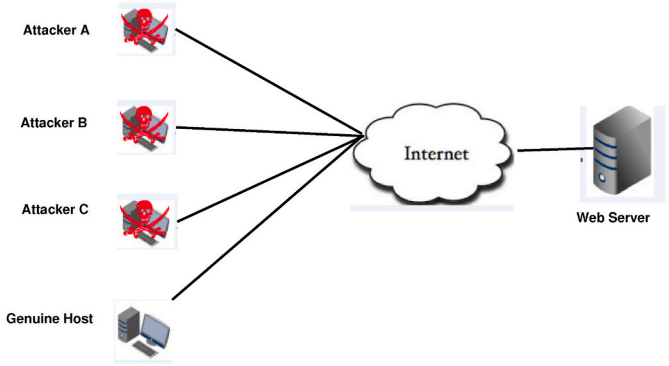

Fig. 1. DDOS Attack: An example.

- Control Plane: The attacker sends a large number of requests from spoofed IP addresses which leads to processing a large number of Packet_in messages at Control Plane. It leads to delay or deny the request of the legitimate user.
- Communication Link between Control Plane and Data Plane: The attacker may attack the communication link between the control and data plane and may deplete the link bandwidth.
- Data plane: The attacker can attack the data plane by flooding the flow table maintained at the switch which leads to flow-table overflow attack

The DDoS attacks can cause heavy damages very easily. For example, In Europe, March 2013, a DDoS attack causes big network congestion against Spamhaus which leads to a huge loss. On Feb. 28, 2018, DDOS attack traffic at 1.3 TBps has crashed the github servers. It also once happened against Amazon web service and disrupted all its services for approximately 2 h and caused a loss of 209 million. From February–March 2019, about 17 DDoS attacks were made on the University of Albany website. The heavy losses incurred from the attack is evident from the above examples. This motivates the author to work on the detection of this attack.

So far, a number of authors (Palmieri, 2019), (Da Silva et al., 2016), (Niyaz et al., 1611), (Santos et al.), (MyintOo et al., 2019), (Cui et al., 2016) have worked for DDoS attack detection in SDN. But these work mostly used unrealistic topologies. Others have used traditional dataset for detecting the attack on SDN. However, some have used the SDN testbed but have not made the data public to validate the approach. The cause for the inapplicability of traditional methods in SDN is due to the architecture difference between two networking paradigms. This motivates the author to work on the SDN testbed and create the SDN traffic Dataset.

1.1. State-of-the-art VS proposed method

At present, the work on DDoS attack detection in SDN is already addressed. But the selection of the significant features which play an important role in attack detection is done in the proposed method. Besides, a hybrid method of Random-forest with Support-vector-classifier is used for the classification task in the proposed work and provides promising results. Preliminary work has been published in Ahuja et al. (Ahuja and Singal, 2019), which incorporates a statistical approach for the detection and prevention of DDOS attacks in SDN. The work has been extended with the machine learning approach in this paper. Several significant works for DDOS detection has been done using Machine learning and deep learning (Ahuja et al., 2021). However, the proposed work differs significantly from them in the following ways.

- Palmieri et al. (Palmieri, 2019), Wang et al. (2015b), latah et al. (Latah and Toker, 2018) have worked on DDOS attack detection but did not use the SDN emulated traffic dataset. Indeed they used a publicly available dataset which has been generated for traditional network architecture and has different set of features that is not applicable in SDN environment.
- Da Silva et al. (Da Silva et al., 2016), Buragohain et al. (Buragohain and Medhi, 2016) have worked on the DDOS attack detection using machine learning approaches where they have considered just two features which can not justify the accuracy achieved whereas the proposed method in the paper used twenty-three features.
- Niyaz et al. (Niyaz et al., 1611) worked on the DDOS attack detection using the Deep Learning technique but the paper has a few limitations in terms of processing capabilities as it has considered 67 features. We have to optimize the number of features to make the system efficient. The proposed work has used 23 features and attained higher accuracy.
- Santos et al. (Santos et al.) have attempted to work on their dataset but they do not follow the real scenario as the dataset is prepared by labeling the traffic serially which means normal traffic followed by attack traffic. But in real-time the attack happens in between the normal traffic and the created dataset in the paper reflects the same scenario.
- Myint et al. (MyintOo et al., 2019), Cui et al. (2016) have claimed to work on their datasets but have not made it public for verification but in our paper, we have shown the results tested on our dataset and made it public as well.
- None of the papers cited above used the hybrid model of SVC-RF which is used in the proposed approach and attained significant results.

The paper proposed the novel dataset which includes the significant features to detect the attack. The proposed approach in the paper is composed of two modules. The first module collects the flow and port statistics to create the dataset and the second module applies a machine-learning algorithm to classify the traffic. The contribution of the paper can be summarized as under:

- **Creation of SDN dataset using mininet emulator**: In the paper, the authors have created the SDN traffic dataset using mininet emulator (OctopressMininet emulation Software, 2018). The datasets which are already available like NSL-KDD, KDD-cup99, ISCX are created for the traditional network and only contain a subset of features available in SDN. So, the author has created the SDN dataset generated in the mininet emulator. The dataset has been shared in the Mendeley data repository given in footer .[1]
- **Classification of traffic into benign and malicious using Machine Learning**: After we create the SDN traffic dataset, different Machine Learning algorithms (Chaudhary et al., 2018) are applied to classify the traffic into benign and malicious class. This trained model that classifies the traffic can also be used in real-time. As the dataset is created on the SDN emulator, the algorithm which produces the highest classification accuracy will also produce similar results in the real-time scenario.
- **Detection of the attack on the host**: To detect the attack on hosts the effective features in the dataset are analyzed by the machine learning model and detect the type of traffic. The rate of normal and attack traffic is kept similar which made the problem of classification more difficult. But it also proves the validity of the features analyzed.

The rest of the paper is categorized into different sections as follows, Section 2 discusses Related Work and discussion of machine-learning models, Section 3 illustrates the DDOS attack and Proposed Methodology, Section 4 illustrates the Experimental Setup and Results and finally Section 5 ends in the conclusion of the paper.

[1] https://data.mendeley.com/datasets/jxpfjc64kr/1.

2. Related Work

In this section, various state-of-the-artwork for detection and prevention of DDOS attack is discussed. Some of them are machine learning-based solutions and others are statistical (Kalkan et al., 2018) based approaches for attack detection. Buragohain et al. (Buragohain and Medhi, 2016) work on the detection and prevention of Distributed Denial of Service attack by setting a constraint on the number of requests made per user. Two features like traffic rate and the total time of a flow duration in the switch are analyzed and their upper and lower limit are recorded for a genuine user. The algorithm analyzes the two features and classifies the traffic. Lower and upper value is defined after analyzing the usage pattern of a normal user. The set can be defined as a tuple:

(Min_traffic_rate, Traffic_rate_max, time_min, time_max). Incoming traffic has to follow the above bounds to be classified as benign traffic otherwise it will be considered malicious. Attack is prevented by reporting to the controller application the lower and upper limits. Latah et al. (Latah and Toker, 2018) used the NSL-KDD dataset and evaluated the various Machine Learning algorithms in classifying the attacks into one of the possible classes. It is a multi-class classification problem. The various classifiers used include Decision Tree, Naive Bayes, Random Forest, Neural Network, K-Nearest Neighbor, and Bagging trees. The number of attack classes in the dataset are

a) Denial of Service b) Remote to Local c) User to root d) Probe attack. All the features present in the dataset are not SDN specific, so the author has three options: (a) To extract a part of the dataset. (b) Find some new features based on the one present in the dataset. (c) To only use control information provided. The author followed the first approach. There are many features available in the dataset but nine features are used. Principle Component Analysis (PCA) has been applied to transform the large dataset into a smaller one. Out of the many classifiers, Neural Network achieved the highest results.

Wang et al. (2020) apply the tensor-based method for DDOS attack detection. Tensors and Eigenvectors are collectively known as Eigen tensors. Tensors are multidimensional arrays where their constituents are defined as a tuple (a1, a2, a3, a4, …,a_n). Tensors are utilized with higher-order data. The decomposition method is used to lower the dimensionality of the data. Tensors are represented in the graph in the form of vectors. Tensor multi-mode product is represented as

$$A_M * X = \lambda * X \tag{1}$$

Here A is a tensor defined as A $\in$ $R^{I_1, I_2, I_3, \ldots, I_m}$, X is the eigen tensor of A and λ is the eigenvalue of A. Data in SDN has been divided into three types namely basic data, parsed data, and instruction data. All the data can be divided based upon their source like flow status information, security information. The data-driven tensor framework for network attack detection uses local tensors to represent data (Source IP, destination IP, source port, destination port, time, number of packets, number of bytes, protocol, network topology data) in SDN. Through tensor operations data from heterogeneous sources are fused to a unified form. Flow table data is also represented as local tensors like matrices which are compressed and later sent to the controller. The tensor data of SDN is prepared by using three features namely Source IP, Destination IP, and Time. The construction of benign and malicious vectors is done using the above features. A squared prediction error is computed and compared with the threshold parameter of Q-statistics. If the value of Q-statistics is greater than the threshold, the attack is detected.

Rasool et al. (2019) apply the concept of Machine learning to detect Link Flooding attack. The author sends the traffic at two different rates to perform slow and fast flooding attack. Burst Header Packet (BHP) flooding attacks dataset is used. Out of the twenty-two features present in the dataset, Fourteen features were used. Experimental results show that Multi-layer perceptron (MLP) attains the best results. The proposed methodology starts by using the Mininet network and traffic is sent across the network. The Openflow statistics are extracted using the python API. Machine learning algorithms start with data pre-processing and processed data is fed to the classifier which finally classifies the traffic.

Da Silva et al. (Da Silva et al., 2016) explained the Distributed Denial of Service attack detection by applying various machine learning algorithms. It used only one feature i.e. flow count present in the switch. When the attack begins the number of flows along with memory consumption also increased at a high rate. During analysis, it was found that when normal traffic is sent the time it takes for a packet to reach the destination is 1.07s. But when the attack takes place the time reaches 1.28s. The memory capacity also increases from 32 KB to 600 KB when the attack takes place. Experimental results show that SVC presents an accuracy of 88.7% and a precision of 82.3%. The simulations were done 35 times until the error was reduced to 0.01 or less.

Ambrosin et al. (2016) work on detection of buffer saturation attack by denying the access of network to the probably blacklisted hosts. Line Switch is implemented as a module at the control plane which protects the controller from attack and when compared to the state-of-the-art algorithms it has reduced the time overhead by 30%. During the process of TCP handshake, the switch acts as a proxy, and the controller is not concerned about any new request. But in this method, a state has to be maintained for each connection at the OpenFlow switch. This generates a new attack called buffer saturation attack. But the available ports on the proxy switch limit the number of TCP connections.

2.1. Machine learning techniques used

Machine Learning algorithms can learn from the features in a dataset.It is known as training the machine learning model. The trained model can classify the traffic into benign and malicious classes. The trained model can also be deployed in real-time to classify the traffic. Some models which perform well in classification tasks as per the literature are mentioned below:

- Logistic Regression (Archer and Lemeshow, 2006): A classification algorithm in Machine Learning which uses a sigmoid function to classify the input into one of the possible classes. It uses a characteristic equation of the form:

$$P(X) = e^{b0+b1*x} / 1 + e^{b0+b1*x} \tag{2}$$

Here P(X) is the probability of X which is the output, b0 is bias and b1 is the coefficient associated with input value (x). Classification label is set to 0 when the probability of an event is less than a given threshold value else it is assigned class label 1. The algorithm is mostly used in the cybersecurity domain for classifying the cases as fraudulent or not. Similarly, the author has tried to predict the likelihood of traffic as Benign or malicious. Besides, the algorithm can also be used for multinomial classification problems. This algorithm produces 83.6% accuracy with the proposed dataset. It is mostly used in the literature, so we also tested the model on our dataset.

- Support Vector Classifier (Amari and Wu, 1999): The algorithm can be used for both regression and classification tasks. But mostly it works well for classification tasks. Various data points are visualized and a decision boundary or plane segregates the class of data points apart. The decision plane can be a straight line or a higher dimensional plane, depends on the number of data points. The points nearest to the decision plane are known as support vectors which help to calculate the margin of the decision plane. Decision plane with large margin are better than small margin. The optimal decision plane is a generalized decision plane and best segregates the different classes. The decision to apply it to our problem is because it works well for classification problem with a large number of features as in our case. But it does not give satisfactory results with our dataset as the features present in our dataset are highly correlated and finding a

decision boundary with such features is not possible. This model, when we train on our dataset yields an accuracy of 85.8%.

- K-Nearest Neighbor (Tan, 2005): It is an unsupervised learning algorithm that works on the principle that similar things exist together. The algorithm works by predicting the label of the new test data by calculating the distance between the test data and other neighbor training samples. The distance is calculated by using the equation mentioned below:

$$distance_{a,b} = \sqrt{(x_2 - x_1)^2 + (y_2 - b_1)^2} \quad (3)$$

This distance is known as Euclidean distance and (x_1, y_1) & (x_2, y_2) are coordinates of two points in space. Other distance measures can also be used like Manhattan Distance, Minkowski distance. When the distance between different neighbors and test data point is found to be minimum, the particular neighbor is selected and its label is assigned to the test sample. This algorithm, when trained with our dataset attains an accuracy score of 95.22%. This algorithm produces significant results as it uses the statistical parameter of distance measurement to compute the similarity.

- Random Forest (Kulkarni and Sinha, 2012): In this method, different decision trees are trained on the dataset. It outputs a class that is the majority vote of the various decision trees. A large number of decision trees are used for final classification results.

A large number of decision trees participate in decision making, the method is also known as an ensemble of decision trees which is shown in Fig. 2. It shows that final output will be based on majority votes of the participating decision trees. It is well suited for a dataset that has a large number of features. This algorithm when trained on our dataset yields an accuracy of 97.2%. Random forest provide accurate predictions as random features are selected during training to each decision tree. Our dataset has a large number of features which increases from 23 to 67 after preprocessing phase and random forest produces significant results with large number of features. We also apply k-fold cross validation during splitting to reject overfitting.

- Ensemble Classifier (Zhou et al., 2020): This classifier considers several models for the task of classification. In this method, the votes of different classifiers are considered as shown in Fig. 3. Figure shows that different classification models (CM_1,CM_2, …CM_n) were considered and trained on the dataset but the final output is decided based on the maximum votes for a class. For eg: if there are three classifiers A, B, and C. A, B predicts the class label as 1 and C predicts class label as 0. The final decision will go with majority votes i.e class 1. The various classifiers include Random Forest, Decision Tree, KNN and SVC. This method attains an accuracy of 97.5% as shown in Fig. 15. This model produces significant results on our dataset as the classification error is reduced by averaging the error of the different classifiers used.

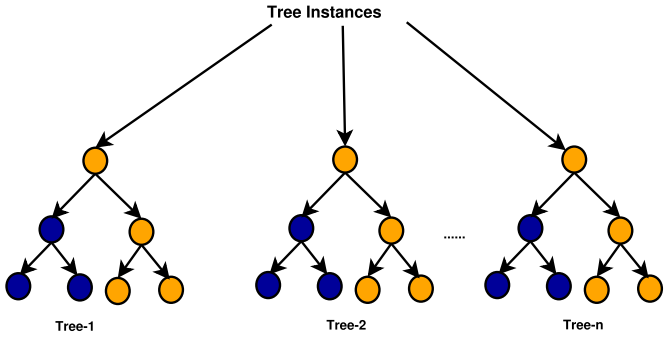

Fig. 2. Random forest.

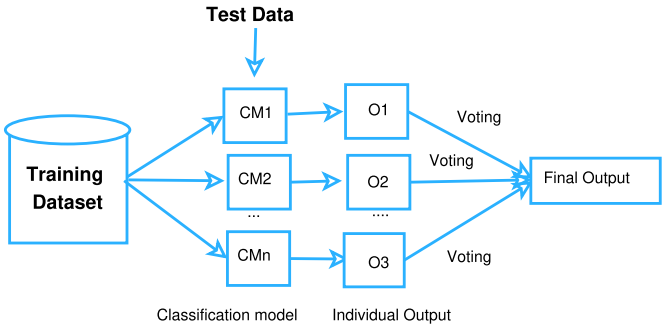

Fig. 3. Ensemble classifier.

- Artificial Neural Network(ANN): It is a network of neurons that duplicates how humans think and reason. It consists of many layers a) Input Layer which consists of a set of input neurons. b) Output layer which consists of a set of classes to which input neurons are mapped. c) Hidden layer consists of computations for fine-tuning the weights in the input layer to minimize the error. The inputs from the input layer are passed to the hidden layer. Each connection is assigned weights and each weight gets multiplied with input neurons and bias is added to them as per the equation below:

$$I = \Sigma_{i=1}^{i=n} w_i * x_i + b \quad (4)$$

The value of I is passed on to the activation function to select the neuron for feature extraction. The same process follows for other hidden layers and the output layer gives the class probability as output. This method achieved a significant accuracy of 98.2% as shown in Fig. 15. But this is a black-box approach to attack detection. The random choice of hyperparameters tuned resulted in a significant accuracy. The same hyperparameter chosen may produce different result on different SDN platform.

- Hybrid Machine learning Model (Support-vector-classifier and Random Forest): In this model, one or more machine learning models are combined to overcome the disadvantage of one-another. The performance of individual classifier is compared with the hybrid model which produces better results. The classification decision by SVC results in some points misclassified near the hyperplane separating the classes. The result of SVC classifier contain both correctly and erroneously predicted results. These are further processed by Randon Forest classifier which act as secondary classifier. Fig. 4 shows the TCP data classified with Linear SVC only. Firstly, the data is reduced to 20 dimensions by using Principle Component Analysis (PCA), a dimensionality reduction algorithm. It is further reduced to two-dimension by T-Stochastic Neighbor embedding (T-SNE) (Wattenberg et al., 2016). Data with unique protocols is fit to separate linear SVCs. But for the points which cannot be inferred to lie in any one of the class are classified by using the inference from Random forest.

3. Distributed Denial of Service attack

In the DDoS attack (Kiwonet al., 2017) presented in Fig. 1, the goal of the attacker is to overflow the resources of the target host to interrupt the benign host. Attackers performed the following attacks:

- Application Plane attack: The attack occurs through the applications present in the application plane. The malicious application exhausts the resources so that legitimate user suffer.
- Controller attack: When the attacker sends a large number of Packet_in requests, the Controller will not able to process all the requests and break down. Subsequently, all the requests from benign user suffer.

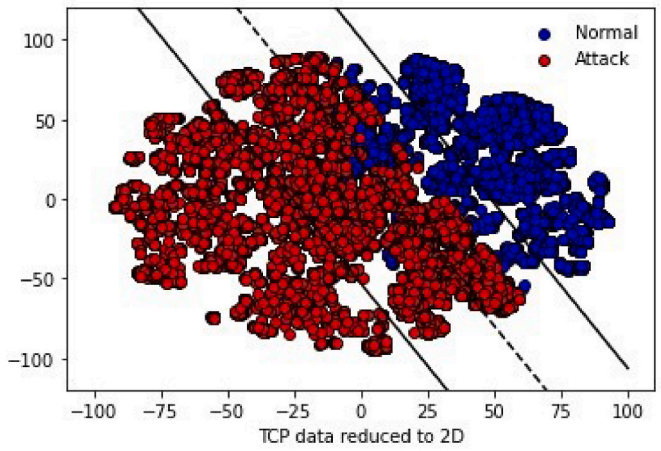

Fig. 4. TCP data classified with Linear SVC.

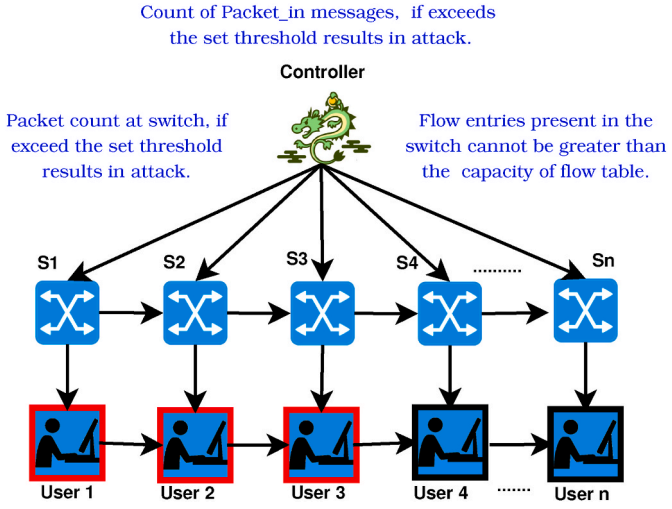

Fig. 5. System model satisfying the constraints.

- Communication link between Control Plane and Data Plane: The attacker may attack the communication link between the control and data plane and may deplete the link bandwidth by sending a large number of Packet_in requests.
- Table-Overflow attack: When the attacker from spoofed IP sources fill the flow table of the switch to its capacity, the benign users suffers. As, when a benign user sends traffic, it will be denied of the services.

3.1. System model

In this section, we are going to describe the SDN architecture where the network is modeled as a directed graph G ∈ (V, E), where V is the set of nodes and E is the set of connections that connects various nodes including hosts, switches, and controller in the network. The problem statement is described by the following attributes:

- Input Data- The input dataset contains seven tuples: {C, S, t, θ, β, ζ, δ} where C is the constraint set that is to be satisfied by the network. S is the set of switches {S_1, S_2, S_3, …, S_n}, t is the time during which statistics are collected, θ & β are the thresholds used, above which the attack is said to take place, ζ is the maximum capacity of emulator and δ is the change in the value of various statistics collected whose value can be analyzed to detect the attack.
- Constraint set (C)- is specified as four tuples C_1, C_2, C_3, C_4 where each constraint can be explained as under:
  - $C1$: $0 <= P < \theta$ Constraint C1 states that the Packet count at each switch should not exceed the threshold value θ. If the packet count (P) exceeds a set threshold value then an attack is said to take place.
  - $C2$: $0 <= P_{in} < \beta$ Constraint C2 states that the count of PacketIn messages (P_{in}) to the controller should be greater than zero and less than a threshold value β. If the number of PacketIn messages exceeds the threshold limit set then the attack is said to take place.
  - $C3$: $D \in 0,1$ Constraint C3 states that decision variable D can take only two values 0, 1 which corresponds to benign and malicious traffic respectively.
  - $C4 : \Sigma_{i=1}^{n} F \leq \zeta$ Constraint C4 states that the number of flow entries (F) present in the switch should not be more than the processing capacity of the emulator(ζ). The System model can be explained by Fig. 5. It depicts the specified constraints are satisfied by different hosts. The various threshold values in the constraints, model the traffic behavior. If the traffic remains within the threshold value, the traffic is recognized as normal and if it exceeds the threshold values the traffic is recognized as malicious. We have generated the normal and attack traffic at a fixed rate of 450 packets per second. So threshold of Packet count above this value can be set to check the data behavior. Similarly, in case of attack traffic the count of the Packet_in mesages exceeds the count for the normal traffic. In an attack scenario a large number of requests are made from spoofed IP addresses which results in a larger number of Packet_in requests to the controller. Also, D is a decision variable that takes either of the two values of 0 or 1. All these constraints are set from the analysis of the dataset.

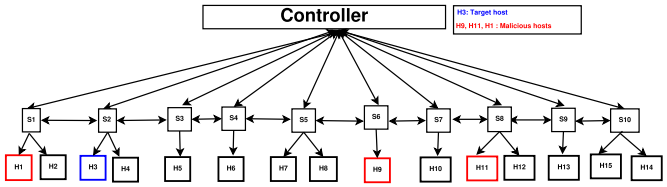

Fig. 6. Topology 1 utilized for dataset creation.

3.2. Dataset creation

Dataset creation is done by extending the controller with a python application which is created with the help of RYU API (Natarajan, 2014b). It monitors the switches present in the topology and collects the various flow and port statistics at a regular monitoring interval. It also writes the collected statistics into a CSV file. Sample topologies used are shown in Figs. 6–8 which show different number of hosts connected with different benign and malicious hosts in each topology.

As the dataset is emulated, it does not contain any missing data. So, the total number of records is 1,04,345 and, each record consist of 23 features. In the dataset the time field represents the date and time at

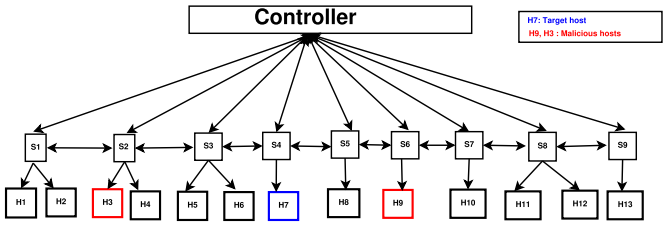

Fig. 7. Topology 2 utilized for dataset creation.

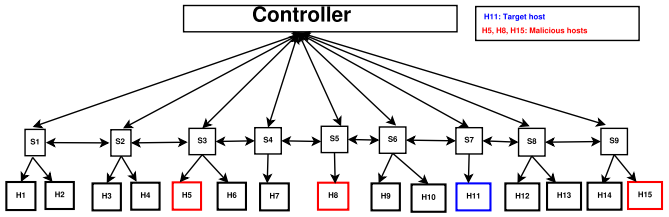

Fig. 8. Topology 3 utilized for dataset creation.

which the data is recorded, Switch represents the datapath-id in the topology, src represents the Source IP Address, DST represents the Destination IP Address, Pktcount represents the count of packets sent during a flow, byte count represents the count of bytes sent during a flow, duration represents the time during which the flow remains in the switch, dur_nsec represents the duration in nanoseconds during which the flow remains in switch and total duration is the total sum of dur_sec and dur_nsec, packetins represents the count of packet_in messages conveyed to the controller, port_no represents the port number of the switch to which the requests are sent, tx_bytes represent the count of bytes sent on a specified switch port, rx_bytes represent the number of bytes transferred to a switch port, tx_kbps represent the kilobytes transferred per second, rx_kbps represent the kilobytes received per second and tot_kbps represent the bandwidth of a switch port.

3.3. Dataset annotation

The annotation of the dataset is done automatically by applying some code logic. The coding is done in such a way that when benign traffic runs, the label column of the dataset is set as "0" and for malicious traffic label is set to "1". After annotation of the data, we apply any machine learning algorithm to classify the traffic. The two classes of the traffic are a) Benign b) Malicious which corresponds to 0, 1 respectively. The total count of traffic instances present in the dataset is depicted in Table 4.

3.4. Feature description

This section discusses the significant features present in the dataset and utilized by a machine learning algorithm to classify the traffic effectively. The various features are critically analyzed and explained below.

- Average Packet count per flow (APPF): In SDN, an attacker take advantage of the fact that source IP is used as a distinguishing feature for identifying a flow. So, the attacker utilized spoofed IP addresses and flood the flow table with different IP addresses. The attacker only aims to flood the flow table and not send data packets. So, the average packet count per flow is a significant feature. APPF decreases in case of attack and increases in case of the benign user as the number of packets sent in attack in less. APPF is the fraction where numerator is the packet count during a flow and denominator is the count of flow entries present in the switch at a particular time. APPF can be expressed as under:

$$APPF = \Sigma_{i=1}^{i=n} c_i / f \tag{5}$$

Here, c is the Packet count per flow which is the total packet sent during a flow and i ranges from $flow_1$ to $flow_n$ and f is the total number of flow entries present in the switch. This feature can be used to detect the DDoS attack.

- Average Byte count per flow (ABPF): In SDN, average byte count per flow can also be used to indicate the attack. This parameter is calculated from one of the flow statistics i.e. byte count. It is calculated in the same way as for APPF. ABPF also decreases in case of attack. ABPF is the fraction where numerator is the byte count during a flow and denominator is the number of flow entries present in the switch. ABPF can be expressed as below:

$$ABPF = \Sigma_{i=1}^{i=n} b_i / f \tag{6}$$

Here, b is the total bytes during a flow. This feature can also be used to detect the DDoS attack.

- Total number of flows in a switch: A flow is defined as the transmission of packets between sending and receiving host in the network. A flow table is maintained at the switch that stores all the flow details for the topology. The total number of flow description in the switch can be expressed as below:

$$f = length(flowtable) \tag{7}$$

Here, flowtable represents the table maintained at the switch whose length gives the number of flow description at a particular instant of time.

In case of attack, the switch connected to the target host is found to contain the maximum number of flow entries. The periodic monitoring of the hosts shows that the target switch has a maximum number of flow entries. So, the number of flow entries is an important feature to consider for classifying the traffic as benign or malicious.

- Protocol: Protocol defines the protocol associated with the traffic flow. Protocol can help identify the traffic protocol associated with the malicious traffic. Any deviation from the normal traffic protocol can help detect the attack.
- Duration: Duration depicts the duration of flow entry in the flow table of a switch. Duration is computed by the sum of d and (d_*nano*). Total duration during which the flow remains in the switch's flow table is a sum of duration_sec (d_*nano*) converted into nanoseconds and duration_sec (d). Total duration in nanoseconds can be expressed as below:

$$Total\ duration = ((d{*}10^9) + (d\_nano)) \tag{8}$$

Attack traffic remains for larger duration as compared to benign traffic. So, the duration during which a flow remains active in the switch plays an important factor in deciding the malicious host. Duration has a high value in case of attack as compared to benign traffic.

- Number of Packet_in messages: Packet_in messages are generated by the hosts and sent to the controller when it does not have a matching flow description in the flow table. In response, the controller sends the Packet_Out message. When the attackers used spoofed IP addresses, a large number of Packet_in messages are reported to the controller. So, the count of Packet_in message is an indicator of an attack. When the Packet_in message is created by the switch it is known as an event. Event handlers are associated with every event generated. So, to obtain the number of Packet_in messages generated, a counter is updated in the Packet_in message event handler routine. Count of Packet_in messages increases in case of an attack. So, it can be used as a significant feature for attack detection.
- Packet rate: Packet rate is defined as the number of packets sent per second. Benign and malicious traffic, both send traffic at the rate of 450 packets per second but as the attacker used spoofed IP addresses, decreases in the packet rate is reported. It can be expressed as below:

$$Packet\ Rate = p_{t+1} - p_t / i \tag{9}$$

where p_{t+1} is the number of packets sent in a flow at time t+1 and p_t is the number of packets sent at time t and i is the monitoring interval. As the packet count is a cumulative number, so packet rate is calculated by subtracting the consecutive packet count and dividing by i.

- Port Bandwidth: Port bandwidth is defined as the sum of received bytes rx_bytes (r) and transmitted bytes tx_bytes (t). The statistics are collected as port statistics from the switch periodically at a regular interval. As the attacker sends more requests and less data, so the Port bandwidth is less which indicates the attack. Port bandwidth is expressed as below:

$$Bandwidth = (t*8)/1000 + (r*8)/1000 \quad (10)$$

Here tx_bytes and rx_bytes are the extracted Port statistics from the switch. The Port bandwidth is calculated by using them and specified in kbps.

3.5. Design of proposed method

This section discusses our proposed methodology. The proposed algorithm 1 shows the step-by-step process of dataset creation. The algorithm collects the flow statistics from the switches present in the topology. For every event there is an event request and reply handler, so during the monitoring interval flow statistics are retrieved by calling OFPFlowStatsRequest method. In reply the event reply handler i.e. OFPFlowStatsReply method is evoked which returns the flow statistics. Similarly, Port statistics are retrieved by calling OFPPortStatsRequest method. These statistics are written to CSV file during the monitoring interval for all the topologies and the dataset is created.

Algorithm 1
Algorithm for creation of Dataset of SDN Traffic

Input: Traffic statistics at switches. Flow and Port statistics are extracted from the switches at a regular interval of 30 s.
Output: SDN traffic Dataset after appending all the flow and port statistics.

Initialization: Normal traffic Packet per flow, Attack traffic Packet per flow, Count of packet_in messages generated, Count of flows generated, Packet Rate.
1: For each flow, collect the flow and port statistics. Out of which some are calculated entries which are shown below:
2: **for** $i = flow_1$ to $flow_n$ **do**
3: Collect all the flow and port statistics from the switches into a file to create the dataset.
4: Annotate the Dataset with Attack traffic as 1.
5: Annotate the Dataset with Normal traffic as 0.
6: **end for**
7: **return** Annotated Dataset.

Fig. 9 shows the process of dataset creation. Here, stepwise phases of dataset creation are shown. After creating and annotating the dataset, machine learning algorithm are trained and tested on the dataset. Fig. 11 shows the process of training and testing the machine learning model on the dataset. Before splitting the dataset, various pre-processing techniques have been applied to the dataset. Dataset is explored by plotting various univariate and bivariate plots to understand the relationship between the independent and dependent variables. Pre-processing techniques applied include handling missing values, null value removal, categorical value encoding, etc.

Various categorical features are present in the dataset such as Source_IP, Destination_IP that needs to be encoded. Other features require normalization in which very high valued features and low

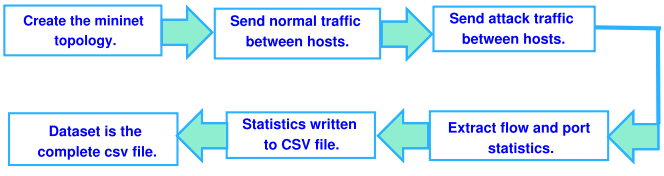

Fig. 9. Block diagram- Dataset Creation.

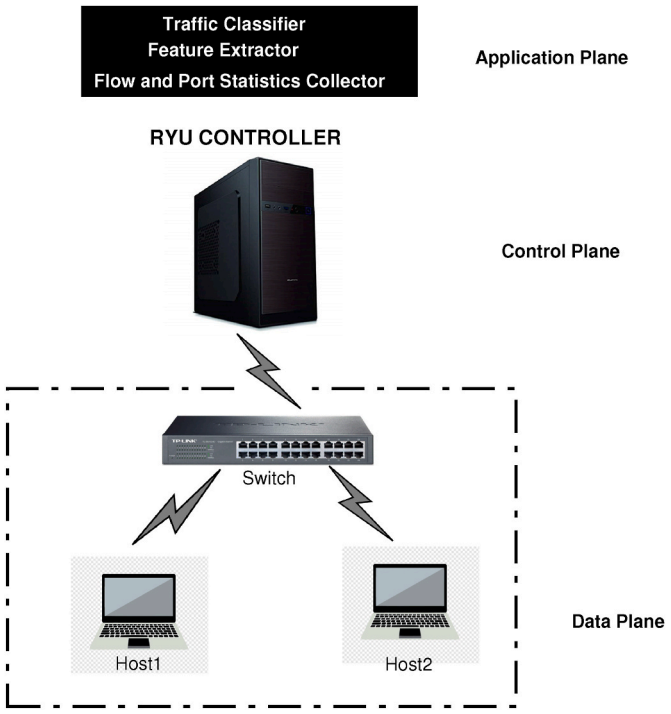

Fig. 10. Experimental setup.

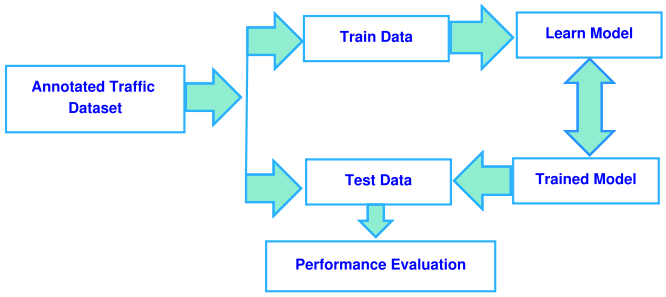

Fig. 11. Classification Using Machine learning.

valued features are scaled to values between (0,1). Before preprocessing, the original dataset comprised 23 columns and 1,04,345 rows. But after preprocessing the dataset comprised of 67 columns due to the addition of dummy variable encoding for categorical variables present in the dataset. The train and test split are done in the ratio of 80:20 and the distribution of the normal and attack traffic present in the dataset is shown in Table 4.

4. Experimental work and results

The work has been carried out on HP EliteBook with 16 GB RAM and 64-bit processor on windows 10. Fig. 10 shows the simulation environment where the single Ryu controller is connected to the openvswitch and the vswitch is connected to the various hosts. Here, sample two hosts are shown, but the topology used is shown in Fig. 6. Benign traffic is generated from random hosts with the help of mgen tool at the rate of 450 packets per second. The benign and malicious traffic are generated with a specific packet size for a specific duration and mentioned in Table 2. The traffic statistics are collected and written to CSV file. However, the malicious traffic is generated from spoofed IP addresses at the rate of 450 packets per second. The spoofed IP address are used by the attacker at the same packet rate as the benign traffic. The benign and malicious traffic are generated in successive batches by

Table 1
Features used in the dataset.

S.No	*Features Used*
1	Packet Count
2	Byte count.
3	Total number of flows in a switch.
4	Packet Count per-flow.
5	Byte Count per-flow.
6	Duration_n_secs.
7	Duration_ secs.
8	Total_duration.
9	Source IP.
10	Destination IP.
11	Number of Packet_in messages.
12	Packet Rate.
13	Port number.
14	txbytes.
15	rxbytes.
16	Port bandwidth.

Table 2
Simulation Environment parameters.

S.No	Parameters
1	Host OS: Windows10
2	Guest OS: Ubuntu16.04
3	VirtualBox: 5.1.26
4	Emulator: Mininet
5	Controller: Ryu
6	Number of Controller: 1
7	Number of Switches: 9
8	Number of Hosts: vary depending on topology
9	Protocol Used: OpenFlow
10	Graphical package: MiniEdit
11	Traffic Generation tool: mgen, hping
12	Controller Port Number: 6653
13	Simulation Time: 250 min
14	Statistics collection interval: 30 s
15	Bandwidth plot interval: 30 s
16	Number of topologies: 10

random hosts. The various statistics are also collected from the switches but they show different behavior compared to benign traffic. The various flow level statistics are collected by calling the OFP-FlowStasRequest method in the Ryu application and port statistics are collected by calling the OFPPortStatsRequest method after every 30 s. We choose an experimental threshold for monitoring interval. The chosen value of the monitoring interval is 30 s because it is the optimal experimental value at which we get the lowest false positives. But as SDN supports programming, this can be changed as per the network environment while sending the network traffic. Statistics request messages are handled as events and the response is handled by the corresponding reply handler. OFPFlowStatsReply includes statistics such as Packet count, byte count, duration_sec, duration_nsecs, Source IP, and Destination IP. OFPPortStatsReply includes statistics such as Port number, tx_bytes and rx_bytes. The statistics collected and computed are shown in Table 2. The dataset is created with 1,04,345 rows. Normal traffic is generated at the rate of 450 packets per second using mgen tool (Natarajan, 2014c) and attack traffic is generated using hping3 (Natarajan, 2014d) to attack the target host. A log of the features collected from the switches is created in the CSV file which is accessed as a data repository on Mendeley. Benign and malicious traffic is generated for 1500 s for ten different topologies which is approximately equal to (15000/60) 250 min data and 1,04,345 CSV rows. Each row of the dataset is comprised of 23 features. Code for the dataset creation and traffic classification is shared in the github repository and the link to the repository appears in the footer.[2] Mininet is used as an emulator on which network topology is emulated and traffic is logged.

Benign traffic generates a combination of TCP, UDP, and ICMP traffic. Attack traffic generates the combination of TCP-SYN attack, UDP flood attack, and ICMP flood attack. Benign traffic is generated using mgen tool (Natarajan, 2014c) and malicious traffic is generated using hping3 (Natarajan, 2014d).

4.1. Performance parameters

Different Machine-Learning models have been trained and tested on our dataset. We evaluated the performance of each model by calculating Accuracy, Sensitivity, Specificity, Precision, and F1-score. A sample confusion matrix is shown below in Table 3: Confusion matrix is often used to calculate the performance parameters mentioned above. In a binary classification problem it is defined as a 2*2 matrix which shows the actual and predicted values of the classifier. The four values in the matrix are rather confusing which are explained as below:

a) True Positive t_p: True positive is the value where the model prediction and actual values in the dataset both are positive. i.e. it is the case where the classifier correctly classify the traffic as benign and malicious.
b) True negative t_n: True negative is the value where the model prediction and actual values in the dataset both are negative. i.e. it is the case where the traffic is correctly classified as malicious.
c) False Positive f_p: False positive is the error type where the model prediction is positive and actual values in the dataset is negative. i.e. it is the case where the traffic is incorrectly classified as benign.
d) False Negative f_n: False negative is error type where the model prediction is negative and actual values in the dataset is positive. i.e. it is the case where the traffic is incorrectly classified as malicious.

Accuracy is one of the measures of performance and it is mathematically defined as a fraction where the numerator specifies the sum of true positive and true negative while the denominator specifies the sum of false positive and false negative along with the terms present in the numerator. It is defined by the equation:

$$Accuracy = ((t_p + t_n) / (t_p + t_n + f_p + f_n)) \quad (11)$$

It is the evaluation of the traffic where the classifier correctly predicts both normal and malicious classes.

Recall is defined as the measure of the correct prediction in the dataset. It is also known as Sensitivity. For example, if the recall of a model is 0.11, it means that the model correctly predicts the correct class 11% of the time. It is represented as (R) and defined by the equation below:

$$Recall(R) = (t_p / (t_p + f_n)) \quad (12)$$

The recall is equivalent to detection rate which is defined as the measure of correctly detecting the malicious traffic. Any IDS requires a high detection rate. As seen from the results, the Hybrid model has the best detection rate in our case.

Table 3
Confusion matrix.

	Positive Predictions	Negative Predictions
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative

[2] https://github.com/nisha077/SDN-traffic-classification.

Specificity is defined as the measure of the prediction of the negative class in the dataset. For example, if the specificity of a model is 0.11, it means that the model correctly predicts the negative class 11% of the time. It is defined by the equation below:

$$Specificity = (t_n \,/\, (t_n + f_p)) \tag{13}$$

A high value of Specificity indicates a lower false positive which leads to more accurate results. The proposed hybrid model has a high specificity value of 98.18% which is a significant result.

False Alarm rate (FAR) is an important parameter to measure the effectiveness of any system. It is the measure of the inaccurate classification when the model classifies the normal traffic as malicious. An intrusion detection system desires to keep FAR as low as possible.

$$FAR = (f_p \,/\, (t_p + f_p)) \tag{14}$$

FAR is an important parameter that is kept as low as possible. Hybrid model achieves the lowest FAR of 0.020 which signifies the results.

Precision is defined as the percentage of the model prediction out of the total data values about the positive class. For example: if the precision of a model is 0.5, it means that out of the total predictions the model makes, it is correct 50% of the time. It is represented as P and defined by the equation:

$$Precision(P) = (t_p \,/\, (t_p + f_p)) \tag{15}$$

Precision predicts the fraction of traffic as benign or malicious, which matches its count present in the dataset.

F1 score is defined as the measure where recall and precision both are used. In the case of an unbalanced dataset, we usually calculate F1-score. It is defined by the equation:

$$F1score = ((2{*}R{*}P) \,/\, (R + P)) \tag{16}$$

From Table 6, we can infer that the accuracy achieved by the proposed hybrid model is best. But due to the unequal instances of different classes present in the dataset as depicted in Table 4, another performance parameters also need to be considered. Other parameters like Precision, recall, and F1-score are also computed in which a high F1-score implies significant results. Hybrid model has attained the best results.

4.2. Result analysis

Evaluation of the system performance is done using the dataset created in Table 6 by calculating Accuracy, Precision, Recall, FAR, and F1-score. Accuracy comparison of SVM-RF hybrid model is done against other specified machine-learning models, shown in Fig. 15. Data exploration is done to understand the data, the number of instances in each class of traffic, Normal and malicious traffic distribution in the dataset. After analyzing the dataset, it can be briefly summarized in the tabular format below. Table 4 specifies the instance count of different traffic classes present in the dataset. Table 5 gives the count of the benign and malicious traffic in each traffic class. This summary data helps to comprehend the dataset. Fig. 12 shows the distribution of the Source IP address used in the dataset. It reveals that the IP addresses used for sending benign and malicious traffic are picked from the same pool. It means IP address alone cannot be used as an indicator of benign or malicious traffic and also significantly use the various features present in Table 1 to detect the attack.

Table 4
Number of instances in the dataset.

Traffic class	Number of instances
Benign	63561
Malicious	40784
TCP Traffic	29436
UDP Traffic	33588
ICMP Traffic	41321

Table 5
Traffic category of each traffic instance.

Traffic class	Benign	Malicious
ICMP	24957	16364
TCP	18897	10539
UDP	22772	10816

Fig. 4 shows dataset has been reduced to two-dimension by the application of PCA and later by t-distributed Stochastic neighbor Embedding(t-SNE). PCA is initially used to reduce the dimensions to twenty. But it only works for the linear dataset and does not work for a non-linear dataset. So, after applying PCA on the dataset, application of t-SNE is done which preserves local distances between the points in high and low dimension and reduced the dimensions to two which also preserve the dataset structure. After the dimensions are reduced, SVC is fit to the dataset and is shown in the figure. Fig. 13 shows the Artificial Neural network (ANN) achieved an accuracy of 98.2%. ANN has considerably high accuracy as compared to other machine learning models but it is a black-box approach to attack detection. The random choice of the hyper-parameters which are tuned only helps to achieve this accuracy score. Hyperparameters chosen depend on the used SDN setting. Fig. 14 shows the cross-entropy loss decreases with an increase in epoch. It shows that as the number of times the model is trained on the dataset, the error in classifying the traffic decreases.

Fig. 15 shows the accuracy comparison of different classifiers. It clearly shows that the hybrid SVC-RF model has achieved the highest accuracy in classifying the traffic. Logistic Regression (LR) model provides an accuracy of 83.69%. It does not offer satisfactory results because the model is not capturing the correct linear relationship between the features and so incorrectly predicted the class labels. It is also widely used in the literature.

K-nearest neighbor (KNN) provides a classification accuracy of 95.22%. KNN is a simple classification model which depends on simple calculations. Calculation of euclidean distance between Support Vector Machine (SVM) provides a classification accuracy of 85.83%. SVM also does not offer satisfactory results because it finds the decision plane with the help of support vectors having the largest separation from both the classes. However, the features present in the dataset are highly correlated and a perfect decision boundary without overfitting might not be always possible.

Random Forest (RF) is a classifier that uses different decision trees for final classification. If one decision tree makes a wrong decision then other trees can compensate for the wrong decision. Each decision tree provides the classification result and the maximum votes are considered to suggest the final classification result. Hence, Rf proves to be a better classifier.

Ensemble classifier (EC) is a classifier that uses several classifier to make a decision. The different classifier which are used includes KNN, Decision Tree, Random Forest and SVC. The accuracy achieved by the classifier is 97.5%, which is substantially better than individual classifier performance.

Support-Vector-classifier with Random-Forest (SVC-RF) hybrid classifier: This classifier is used as a combination of two machine learning algorithms and gives the best results with our dataset.

Fig. 16 shows that when the attack takes place the bandwidth consumption increased and reaches in the range of 500–2000 kbps but when countermeasures are applied it reduces and reaches 100–500 kbps. When countermeasures are applied packets are dropped by host 1 (port 1) as the attack is detected. Host 2,3 are still sending the traffic but host 1 drops the packets.

Table 6
Performance Measures of different Algorithms.

Model	Accuracy	Detection Rate	FAR	Specificity	Precision	F1-Score
Logistic Regression	83.69%	82.46%	0.175	83.97%	83.31%	82.26%
SVC	85.83%	87.46%	0.125	84.04%	85.79%	86.61%
KNN	95.22%	94.37%	0.056	92.34%	96.83%	95.58%
Random Forest	97.2%	95.45%	0.045	94.56%	96.56%	96.23%
Ensemble Classifier	97.5%	96.43%	0.036	95.32%	96.43%	96.72%
ANN	98.2%	97.84%	0.022	97.43%	97.43%	97.12%
SVC-RF	98.8%	97.91%	0.02%	98.18%	98.27%	97.65%

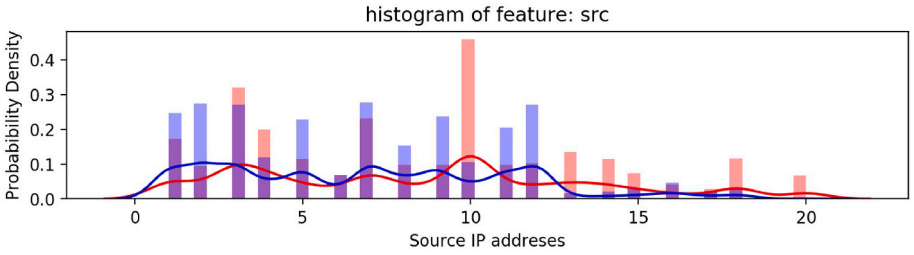

Fig. 12. Distribution of Source IP address in Normal and Attack Traffic.

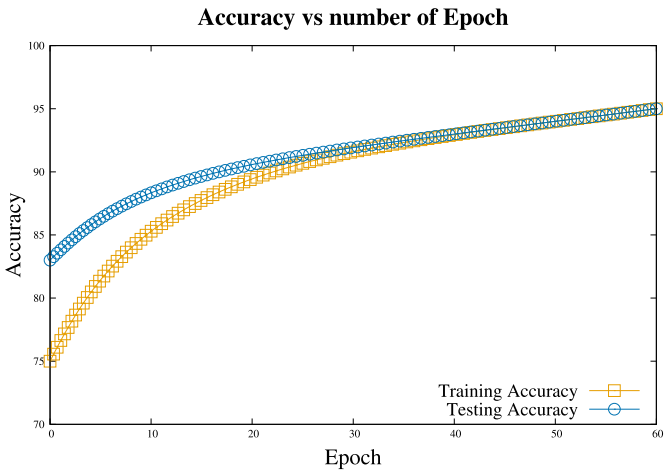

Fig. 13. Accuracy vs number of Epoch.

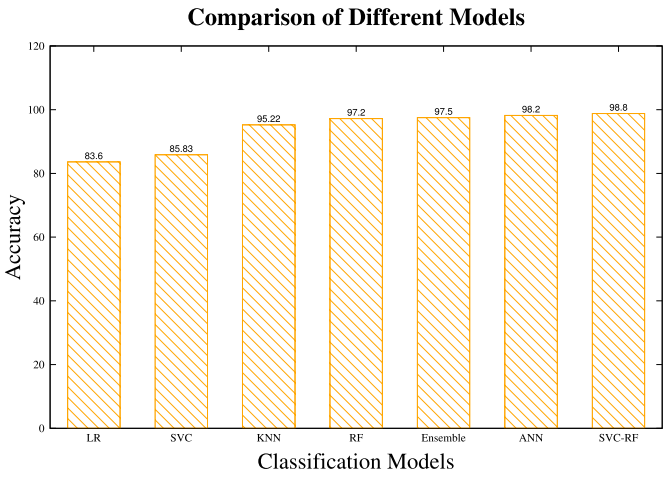

Fig. 15. Comparative analysis of different Algorithms Used.

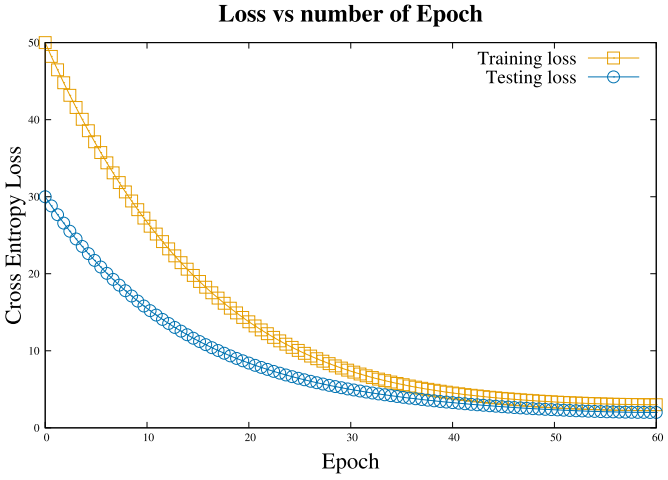

Fig. 14. Loss vs number of Epoch.

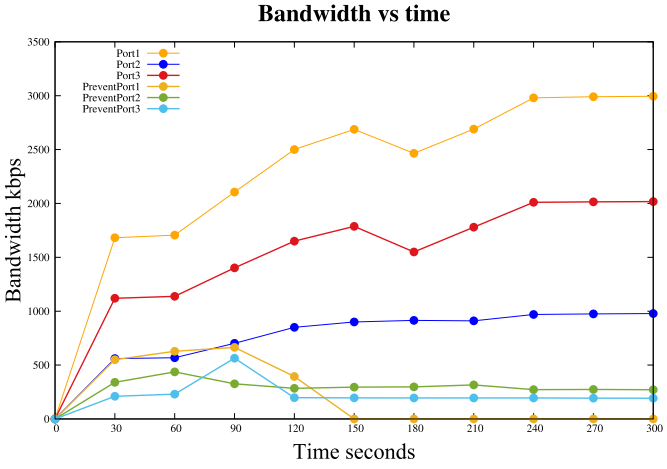

Fig. 16. Consumption of Bandwidth in attacked vs after prevention technique applied.

4.3. Comparative analysis with existing results

To evaluate the proposed method in the paper (Refer to Table 7), a comparison is done with the existing work done in the area of DDOS attack detection using emulated dataset. The top existing benchmark result is found to be 96%. From the Table, it can be seen that our proposed model achieves the highest accuracy of 98.8%. The hybrid approach followed for the proposed model plays a significant role in attack detection.

Dataset for unique protocols is fit to linear Support-Vector-Classifier (SVC) as shown in Fig. 4 and for the suspicious points, inference from the Random forest is used for final classification. This model proved to be the best performing model for our dataset with reduced training time as well.

4.4. Observation and discussion

The results shown above clearly indicate that a network administrator of an organization can easily use in his capacity the machine learning model developed for early detection of the attack. The DDoS attack can be detected by implementing the proposed machine learning model. The experimental work is done on the SDN dataset created on the emulator, the software version of the real switch, so the results will be valid in real-time. The features proposed in the dataset are clearly explained in Table 1. The same set of features can be collected using the proposed method in real-time and the machine learning algorithms can be deployed for classification. The hybrid model of Support Vector machine and Random Forest has produced high precision and recall value which is a promising solution in real-time. SVC-RF model works well as the misclassification of points done by SVC are taken care by RF.

Various machine learning algorithms used show promising results for detecting the attack. The proposed hybrid model shows highest results in terms of various performance parameters used because of the novel and significant features which we have proposed in our dataset. Also, the proposed machine learning technique of SVC filtered by Random forest help find the decision boundary which best fits to the data and achieve significant results. Previous work which has been done does not achieve significant results as shown in Fig. 15, which yet proves the significance of our results. Previously, attack detection is done by usin some statistical approaches, which we also have implemented in our previous paper (Ahuja and Singal, 2019) which involve analysis of the various flow and port statistics. But now with the created SDN dataset, we can use promising technologies of Machine Learning and Deep Learning to classify the traffic. The graph in Fig. 16 is the results of the previous paper which show the effect of the applying prevention technique to the bandwidth. During the attack, the bandwidth consumption reaches in the range of 3000 kbps but when the prevention technique is applied the bandwidth reduces to 600 kbps.

5. Conclusion & future work

Software-Defined Networking (SDN) is the networking architecture defined by software. Network traffic is controlled in a centralized manner by the controller which remotely directs the traffic between the hosts. Apart from such a flexible network control of traffic management, the network is still prone to various attacks. In this paper, the authors work on creating an SDN dataset and classification of the benign traffic from DDOS attack traffic by applying the machine learning algorithms. Machine learning models are trained on the SDN dataset which is created on mininet emulator which resembles the real-time scenario, so the classification model can be used in real-time to classify the traffic based on the learned features. The author proposed the hybrid model of Support Vector classifier and Random Forest which classify the traffic by using SVC results which are filtered again by Random Forest. It achieves the highest accuracy of 98.8%. High accuracy specifies that the detection can be done irrespective of the traffic protocol as the dataset has proposed significant features. With 97.91% sensitivity high true positives are depicted which signifies the performance of the model, 98.27% precision means the model is predicting the class label correctly in most of the cases.

In the future, we will analyse and apply the performance of deep learning models for DDOS attacks along with topology poisoning attacks identification. Application of Deep Learning techniques for classification of benign traffic from attack traffic is computational intensive but provide promising results.

Table 7
Comparison Results of traffic classification using various Simulated SDN Datasets.

S.No	*Authors*	*Testing Accuracy*
1	Metiet al. (Meti et al., 2017)	80%
2	da Silva et al. (Da Silva et al., 2016)	88.7%
3	perez et al. (Pérez-Díaz et al.)	95%
4	Ye et al. (Ye et al., 2018)	95.24%
5	Ko et al. (Ko et al.)	96%
6	han et al. (Han et al., 2018)	96%
7	myint et al. (MyintOo et al., 2019)	97%
8	**Proposed**	**98.8%**

Author contribution

Nisha ahuja does the formulation of the problem. G. Singal does the result analysis of the scheme. D. Makhopadhya provides the literature review analysis of various proposals. N.Kumar does the final editing and problem identification with topology identification of the scheme.

Declaration of competing interest

There is no conflict of interest during the submission of the paper at this venue.

References

Ahuja, N., Singal, G., 2019. Ddos attack detection & prevention in sdn using openflow statistics. In: 2019 IEEE 9th International Conference on Advanced Computing (IACC). IEEE, pp. 147–152.

Ahuja, N., Singal, G., Mukhopadhyay, D., 2021. DLSDN: Deep Learning for DDOS attack detection in Software Defined Networking. In: Data Science & Engineering (Confluence), pp. 683–688. https://doi.org/10.1109/Confluence51648.2021.9376879.

Amari, S.-i., Wu, S., 1999. Improving support vector machine classifiers by modifying kernel functions. Neural Network. 12 (6), 783–789.

Ambrosin, M., Conti, M., De Gaspari, F., Poovendran, R., 2016. Lineswitch: tackling control plane saturation attacks in software-defined networking. IEEE/ACM Trans. Netw. 25 (2), 1206–1219.

Archer, K.J., Lemeshow, S., 2006. Goodness-of-fit test for a logistic regression model fitted using survey sample data. STATA J. 6 (1), 97–105.

Buragohain, C., Medhi, N., 2016. Flowtrapp: an sdn based architecture for ddos attack detection and mitigation in data centers. In: 2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN). IEEE, pp. 519–524.

Chaudhary, R., Aujla, G.S., Garg, S., Kumar, N., Rodrigues, J.J.P.C., 2018. Sdn-enabled multi-attribute-based secure communication for smart grid in iiot environment. IEEE Trans. Ind. Inf. 14 (6), 2629–2640.

Cui, Y., Yan, L., Li, S., Xing, H., Pan, W., Zhu, J., Zheng, X., 2016. Sd-anti-ddos: fast and efficient ddos defense in software-defined networks. J. Netw. Comput. Appl. 68, 65–79.

Da Silva, A.S., Wickboldt, J.A., Granville, L.Z., Schaeffer-Filho, A., 2016. Atlantic: a framework for anomaly traffic detection, classification, and mitigation in sdn. In: NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium. IEEE, pp. 27–35.

Dabbagh, M., Hamdaoui, B., Guizani, M., Rayes, A., 2015. Software-defined networking security: pros and cons. IEEE Commun. Mag. 53 (6), 73–79.

Garg, S., Kaur, K., Kumar, N., Rodrigues, J.J.P.C., 2019. Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in sdn: a social multimedia perspective. IEEE Trans. Multimed. 21 (3), 566–578.

Han, B., Yang, X., Sun, Z., Huang, J., Su, J., 2018. Overwatch: a cross-plane ddos attack defense framework with collaborative intelligence in sdn. Secur. Commun. Network.

Kalkan, K., Altay, L., Gür, G., Alagöz, F., 2018. Jess: joint entropy-based ddos defense scheme in sdn. IEEE J. Sel. Area. Commun. 36 (10), 2358–2372.

Kiwon, H., et al., 2017. Sdn-assisted slow http ddos at tack defense method. IEEE Commun. Lett. 20 (17), 1–1.

I. Ko, D. Chambers, E. Barrett, Self-supervised network traffic management for ddos mitigation within the isp domain, Future Generat. Comput. Syst..
Kreutz, D., Ramos, F.M., Verissimo, P., 2013. Towards secure and dependable software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 55–60.
Kulkarni, V.Y., Sinha, P.K., 2012. Pruning of random forest classifiers: a survey and future directions. In: 2012 International Conference on Data Science & Engineering (ICDSE). IEEE, pp. 64–68.
Latah, M., Toker, L., 2018. Towards an efficient anomaly-based intrusion detection for software-defined networks. IET Netw. 7 (6), 453–459.
Meti, N., Narayan, D., Baligar, V., 2017. Detection of distributed denial of service attacks using machine learning algorithms in software defined networks. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, pp. 1366–1371.
Myint Oo, M., Kamolphiwong, S., Kamolphiwong, T., Vasupongayya, S., 2019. Advanced support vector machine-(asvm-) based detection for distributed denial of service (ddos) attack on software defined networking (sdn). J. Comput. Netw. Commun.
Natarajan, S., 2014a. Ryu Controller. http://www.sdnhub.org/tutorials/ryu/. (Accessed 19 July 2014).
Natarajan, S., 2014b. Mgen. https://ryu.readthedocs.io/en/latest/ryu_app_api.html. (Accessed 31 October 2015).
Natarajan, S., 2014c. Mgen. https://www.nrl.navy.mil/itd/ncs/products/mgen. (Accessed 31 October 2015).
Natarajan, S., 2014d. Hping3. http://www.hping.org/hping3.html. (Accessed 21 July 2016).
Q. Niyaz, W. Sun, A. Y. Javaid, A Deep Learning Based Ddos Detection System in Software-Defined Networking (Sdn), arXiv preprint arXiv:1611.07400.
Octopress, Mininet emulation Software, 2018. http://www.mininet.org/. (Accessed 19 July 2008).
Palmieri, F., 2019. Network anomaly detection based on logistic regression of nonlinear chaotic invariants. J. Netw. Comput. Appl. 148, 102460.
J. A. Pérez-Díaz, I. A. Valdovinos, K.-K. R. Choo, D. Zhu, A Flexible Sdn-Based Architecture for Identifying and Mitigating Low-Rate Ddos Attacks Using Machine Learning, IEEE Access.
Rasool, R.U., Ashraf, U., Ahmed, K., Wang, H., Rafique, W., Anwar, Z., Cyberpulse, 2019. A machine learning based link flooding attack mitigation system for software defined networks. IEEE Access 7, 34885–34899.
M. Said Elsayed, N.-A. Le-Khac, S. Dev, A. Delia Jurcut, Machine-learning Techniques for Detecting Attacks in Sdn Using Nsl-Kdd, arXiv preprint arXiv:1910.00817.
R. Santos, D. Souza, W. Santo, A. Ribeiro, E. Moreno, Machine Learning Algorithms to Detect Ddos Attacks in Sdn, Concurrency and Computation: Practice and Experience e5402.
Shin, S., Gu, G., 2013. Attacking software-defined networks: a first feasibility study. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 165–166.
Singal, G., Laxmi, V., Rao, V., Todi, S., Gaur, M.S., 2017a. Improved multicast routing in manets using link stability and route stability. Int. J. Commun. Syst. 30 (11), e3243.
Singal, G., Laxmi, V., Gaur, M.S., Rao, V., 2017b. Moralism: mobility prediction with link stability based multicast routing protocol in manets. Wireless Network 23 (3), 663–679.
Tan, S., 2005. Neighbor-weighted k-nearest neighbor for unbalanced text corpus. Expert Syst. Appl. 28 (4), 667–671.
Varadharajan, V., Karmakar, K., Tupakula, U., Hitchens, M., 2018. A policy-based security architecture for software-defined networks. IEEE Trans. Inf. Forensics Secur. 14 (4), 897–912.
Wang, H., Xu, L., Gu, G., 2015a. Floodguard: a dos attack prevention extension in software-defined networks. In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. IEEE, pp. 239–250.
Wang, B., Zheng, Y., Lou, W., Hou, Y.T., 2015b. Ddos attack protection in the era of cloud computing and software-defined networking. Comput. Network. 81, 308–319.
Wang, P., Yang, L.T., Nie, X., Ren, Z., Li, J., Kuang, L., 2020. Data-driven software defined network attack detection: state-of-the-art and perspectives. Inf. Sci. 513, 65–83.
Wattenberg, M., Viégas, F., Johnson, I., 2016. How to use t-sne effectively. Distill 1 (10), e2.
Ye, J., Cheng, X., Zhu, J., Feng, L., Song, L., 2018. A Ddos Attack Detection Method Based on Svm in Software Defined Network, Security and Communication Networks.
Yoon, C., Lee, S., Kang, H., Park, T., Shin, S., Yegneswaran, V., Porras, P., Gu, G., 2017. Flow wars: systemizing the attack surface and defenses in software-defined networks. IEEE/ACM Trans. Netw. 25 (6), 3514–3530.
Zhou, Y., Cheng, G., Jiang, S., Dai, M., 2020. Building an Efficient Intrusion Detection System Based on Feature Selection and Ensemble Classifier. Computer Networks, p. 107247.

Nisha Ahuja received the B. Tech degree in Computer Science and Engineering from Kurukshetra University, India, in 2007, and the M. Tech. degree in Computer Science and Engineering from Maharishi Dayanand University, India, in 2012. She is currently pursuing the Ph.D. degree with the School of Engineering and Applied Sciences, Bennett University, India. Her research interests include Network security, Software Defined Networking, machine learning, and deep learning.

Gaurav Singal is an Assistant Professor in Computer Science Engineering Department at Bennett University, Greater Noida, India. He obtained his Ph.D. and M. Tech. in Computer Science Engineering department from Malaviya National Institute of Technology, Jaipur, India. He received the research grants from Department of Science and Technology, Uttar Pradesh on women security and Department of Biotechnology on Assistive devices. He is actively working in research and teaching from last 9 years and published number of reputed conferences and journals. He is the member of scientific society IEEE and ACM.

Debajyoti Mukhopadhyay (SMIEEE(USA), SMACM(USA), FIE (India), FIETE(India)) is Dean of School of Engineering and Applied Sciences at Bennett University, India. Earlier, he had held the position of Dean (R&D) at MIT, Pune, Director and Dean at Mumbai University, Dean at Adamas University. He has been the Founding Director of the Web Intelligence and Distributed Computing Research Lab since 2002. He was a Visiting Scholar at George Mason University, Virginia, USA. He worked as a Professor of Computer Science and Engineering at West Bengal University of Technology. He was a Visiting Professor at Chonbuk National University, Korea. He worked at Bellcore, USA, 1987-94. He has published nearly 230 research articles. He holds a BE (Electronics) from IIEST Shibpur, DCS (Computer Science) from Queen's University Belfast, MS (Computer Science) from Stevens Institute of Technology and PhD (Engineering) from Jadavpur University.

Neeraj Kumar received his Ph.D. in CSE from SMVD University, Katra (J & K), India, and was a postdoctoral research fellow in Coventry University, Coventry, UK. He is working as an Associate Professor in the Department of Computer Science and Engineering, Thapar Institute of Engineering & Technology, Patiala (Pb.), India since 2014. Dr. Neeraj is an internationally renowned researcher in the areas of VANET & CPS Smart Grid & IoT Mobile Cloud computing & Big Data and Cryptography. He has published more than 150 technical research papers in leading journals and conferences from IEEE, Elsevier, Springer, John Wiley, and Taylor and Francis. He has total research funding from these agencies of more than 2 Crores under different schemes from the GOI. He has h-index of 26 (according to Google scholar, March 2017) with 2500 citations to his credit. He is editorial board members of International Journal of Communication Systems, Wiley, Security and Communication, John Wiley, and Journal of Networks and Computer Applications, Elsevier. He has visited many countries mainly for the academic purposes. He is a visiting research fellow at Coventry University, Coventry, UK. He has many research collaborations with premier institutions in India and different universities across the globe. He has supervised 5 Ph.D. students and 5 are currently pursuing their thesis. He has also supervised more than 20 M.E./M.Tech. thesis.